2009

Software Hardware Asset Reuse Enterprise (SHARE) repository framework: related work and development plan / by Jean Johnson, Curtis Blais.

NPS-GSBPP-09-014

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

Software Hardware Asset Reuse Enterprise (SHARE) Repository Framework: Related Work and Development Plan

19 August 2009

by

Jean Johnson, Research Assistant, and

Curtis Blais, Research Associate

Graduate School of Engineering and Applied Sciences

Naval Postgraduate School

Prepared for: Naval Postgraduate School, Monterey, California 93943

THIS PAGE INTENTIONALLY LEFT BLANK

Naval Postgraduate School
Monterey, California

Daniel T. Oliver
President

Leonard A. Ferrari
Provost

The Acquisition Chair, Graduate School of Business & Public Policy, Naval Postgraduate School supported the funding of the research presented herein.

The report was prepared by:

_____
Jean Johnson, Research Assistant
Graduate School of Engineering & Applied Sciences

_____
Curtis Blais, Research Associate
Graduate School of Engineering & Applied Sciences

Reviewed by:

_____
William R. Gates, Ph.D.
Dean, Graduate School of Business & Public Policy

Released by:

_____
Karl van Bibber, Ph.D.
Vice President and
Dean of Research

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form approved
OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	19 August 2009	1 October 2007 through 30 September 2008

4. TITLE AND SUBTITLE
Software Hardware Asset Reuse Enterprise (SHARE) Repository Framework: Related Work and Development Plan

5. FUNDING N/A

6. AUTHOR (S)
Jean Johnson and Curtis Blais

7. PERFORMING ORGANIZATION NAME (S) AND ADDRESS (ES)
NAVAL POSTGRADUATE SCHOOL
GRADUATE SCHOOL OF BUSINESS AND PUBLIC POLICY
555 DYER ROAD
MONTEREY, CA 93943-5103

8. PERFORMING ORGANIZATION REPORT NUMBER
NPS-GSBPP-09-014

9. SPONSORING/MONITORING AGENCY NAME (S) AND ADDRESS (ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words.)

In August 2006, Program Executive Officer, Integrated Warfare Systems (PEO-IWS), established the Software, Hardware Asset Reuse Enterprise (SHARE) repository to enable the reuse of combat system software and related assets. A description of SHARE and the requirements for a component specification and ontology supporting this repository are available in Johnson (2007). The Naval Postgraduate School (NPS) is tasked to develop this component specification and ontology for the SHARE repository. This report gives our vision of the component specification and ontology, while providing a brief survey of initiatives and technologies relevant to desired repository capabilities. We then describe the development approach and initial design of the component specification and ontology. We conclude with recommended next steps for continuing development of the repository capabilities.

14. SUBJECT TERMS
Software Reuse, Software repository, Component Specification, Ontology

15. NUMBER OF PAGES
75

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT: UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE: UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT: UNCLASSIFIED	20. LIMITATION OF ABSTRACT: UU

THIS PAGE INTENTIONALLY LEFT BLANK

Abstract

In August 2006, Program Executive Officer, Integrated Warfare Systems (PEO-IWS), established the Software, Hardware Asset Reuse Enterprise (SHARE) repository to enable the reuse of combat system software and related assets. A description of SHARE and the requirements for a component specification and ontology supporting this repository are available in Johnson (2007). The Naval Postgraduate School (NPS) is tasked to develop this component specification and ontology for the SHARE repository. This report gives our vision of the component specification and ontology, while providing a brief survey of initiatives and technologies relevant to desired repository capabilities. We then describe the development approach and initial design of the component specification and ontology. We conclude with recommended next steps for continuing development of the repository capabilities.

Keywords: Keywords: Software Reuse, Software repository, Component Specification, Ontology

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgements

The authors would like to acknowledge Dr. Mikhail Auguston for his contributions to the vision for the component specification and ontology. We would also like to thank the sponsor, Mr. Nick Guertin, PEO IWS 7B, the SHARE program manager, Ms. Barbara Doyal, and Mr. Mark Wessman for their guidance throughout this research.

THIS PAGE INTENTIONALLY LEFT BLANK

About the Authors

Jean Johnson is a research assistant in the Naval Postgraduate School Systems Engineering Department. She coordinates a program to develop Modeling and Simulation curriculum for DoD Acquisition workforce personnel and performs research for the PEO IWS SHARE program. Previously, Ms. Johnson held various positions supporting NAVSEA's Warfare Systems Engineering Division. She is a US Navy active and reserve veteran and continues her Navy affiliation in the Individual Ready Reserve. Ms. Johnson holds a ME in Operations Research and Systems Analysis and a BS in Applied Mathematics from Old Dominion University and is currently pursuing her PhD in Software Engineering at NPS.

Jean M. Johnson
Systems Engineering Department
Naval Postgraduate School
Monterey, CA 93943-5000
Tel: 831-656-2956
Fax: (831) 656-3219
E-mail: jmjohnso@nps.edu

Curtis Blais is a research associate in the Naval Postgraduate School Modeling, Virtual Environments, and Simulation (MOVES) Institute. His research interests include application of Web-based technologies to improve interoperability of C2 systems and M&S systems. He is contributing to metadata design efforts for the SHARE program, the Department of Defense (DoD) M&S Community of Interest Discovery Metadata Specification, M&S Catalog, and standardized DoD Verification, Validation, and Accreditation (VV&A) documentation, as well as international standardization efforts for the Military Scenario Definition Language and the Coalition Battle Management Language. Mr. Blais hold BS and MS degrees in Mathematics from the University of Notre Dame and is a PhD candidate in the MOVES program.

Curtis Blais
Modeling, Virtual Environments and Simulation Institute
Naval Postgraduate School
Monterey, CA 93943-5000
Tel: 831-656-3215
Fax: (831) 656-7599
E-mail: clblais@nps.edu

THIS PAGE INTENTIONALLY LEFT BLANK

NPS-GSBPP-09-014

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

Software Hardware Asset Reuse Enterprise (SHARE) Repository Framework: Related Work and Development Plan

19 August 2009

by

Jean Johnson, Research Assistant, and

Curtis Blais, Research Associate

Graduate School of Engineering and Applied Sciences

Naval Postgraduate School

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

Glossary of Abbreviations

ADL	Architecture Description Language
ASW	Anti-Submarine Warfare
CIEL	Common Information Element List
CIO	Chief Information Officer
COAL	Common Operational Activities List
COI	Community of Interest
COM	Component Object Model
CPAN	Comprehensive Perl Archive Network
CSFL	Common Systems Function List
DDMS	DoD Discovery Metadata Specification
DISA	Defense Information Systems Agency
DL	Description Logic
DMS	Discovery Metadata Specification
DoD	Department of Defense
GIG	Global Information Grid
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDL	Interface Definition Language
IRI	International Resource Identifier
ISO	International Organization for Standardization
IWS	Integrated Warfare Systems
LCS	Littoral Combat Ship
LSI	Latent Semantic Indexing

M&S	Modeling and Simulation
MDA	Maritime Domain Awareness; also Model Driven Architecture
MDR	Metadata Registry
MOF	Meta Object Facility
MSC	Modeling and Simulation Community of Interest
NCW	Net-Centric Warfare
NDA	Non-Disclosure Agreement
NPS	Naval Postgraduate School
OA	Open Architecture
OCL	Object Constraint Language
OMG	Object Management Group
OWL	Web Ontology Language
OWL-S	Web Ontology Language for Services
PEO	Program Executive Office
RAS	Reusable Asset Specification
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RIF	Rule Interchange Format
SBA	Simulation-Based Acquisition
SHARE	Software Hardware Asset Reuse Enterprise
SIAP	Single Integrated Air Picture
SIPRNET	Secret Internet Protocol Router Network
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol

SPARQL	SPARQL Protocol and RDF Query Language
SSDS	Ship Self Defense System
UDDI	Universal Description, Discovery, and Integration
UML	Unified Modeling Language
URI	Universal Resource Identifier
URL	Universal Resource Locator
USW	Undersea Warfare
W3C	World Wide Web Consortium
WS-BPEL	Web Services Business Process Execution Language
WSDL	Web Services Description Language
WS-I	Web Services Interoperability
XMI	XML Metadata Interchange
XML	Extensible Markup Language

THIS PAGE INTENTIONALLY LEFT BLANK

I. Introduction

A. Background

In August 2006, the Program Executive Officer of Integrated Warfare Systems (PEO IWS) established the Software, Hardware Asset Reuse Enterprise (SHARE), a library of combat system software and related assets, for use by eligible contractors (both prime contractors and subcontractors) for developing or suggesting improvements to Navy Surface Warfare Systems. The SHARE repository is presently being populated. As of January 2008, 62 assets containing 18,017 artifacts from the Aegis, Ship Self Defense System (SSDS), Littoral Combat Ship (LCS), DDG-1000, and Single Integrated Air Picture (SIAP) programs are available in the SHARE library. PEO-IWS encourages all current Navy contractors and potential Navy contractors to register for access to the SHARE library to discover the assets it presently contains, as well as to contribute assets that may be useful to the Navy and its contracting community in the future.[1] An unclassified site (https://viewnet.nswc.navy.mil) provides a mechanism to discover available library materials within SHARE, as those materials are populated. The site also hosts the license agreement and Non-Disclosure Agreement (NDA) required for obtaining library materials. Library materials are provided either online through access to the appropriate portion of the SHARE web site (classified or unclassified) or via delivery of physical media. The registration process for the classified portion of the site over SIPRNET (https://viewnet.nswcdd.navy.smil.mil) is the same as the unclassified portion above, except no digital certificate is required. The SHARE repository is currently being updated to Version 1.3 (March 2008), which incorporates several new enhancements, including updated metadata and an improved asset submission process. A more complete description of SHARE and the requirements for a semantic framework consisting of a component (repository asset) specification and ontology that will support this repository are available in Johnson (2007).

[1] Organizations interested in registering for access to the library should visit and complete an online registration form at https://viewnet.nswc.navy.mil

B. Scope

The Naval Postgraduate School (NPS) is tasked to develop an initial component specification and ontology for the SHARE software[2] repository. The component specification will describe the artifacts contained in the repository in sufficient detail to aid a repository user in determining if the artifact is worth retrieving. The ontology will provide contextual semantics describing relationships among items in the repository to aid in associating artifacts with user needs. The component specification and ontology will comprise a rich structural and semantic framework for SHARE that will enable multiple kinds of search and discovery techniques. The goal is to enable the development of different types of tools to improve the usefulness of SHARE.

C. Purpose

The purpose of this report is to describe relevant existing technologies to the SHARE framework and to identify their utility in the intended framework development approach. We begin by providing a description of the intended repository framework, and then we discuss existing technologies and initiatives that may be advantageous to its development. Finally, we share our proposed approach for the completion of the framework development project.

[2] While the SHARE repository is intended for information on both hardware and software assets, this initial tasking is limited to a software asset scope. The goal of the research, however, is for technical approaches that are developed for software to be readily adaptable to descriptions of hardware assets in the repository.

II. Conceptual Vision for the Software Repository Framework

A. Introduction

In this section, we discuss typical current repository practices, outline our vision for improving upon them, and describe our overall approach for the SHARE research project.

B. Repositories Today

Software repositories today tend to be organized to support keyword searches over broad categories of software types. They vary greatly in the amount of information (metadata) available for each artifact in the repository.

1. SourceForge

One of the most popular online open source repositories, SourceForge is both a software repository and a project management tool. The project management portion of the site requires registration of a project and enables coordination of configuration management, task management, and other project communication concerns for development projects. The repository contains downloadable software from the projects— some are free and some are provided at a cost.

The SourceForge repository enables essentially two different ways to search. First, users can browse the repository by clicking through categorizations of different types of software and then refine the search by filtering for different program aspects, such as specific program language or operating system. Second, a keyword search over the metadata within a particular category is possible. The metadata in SourceForge is quite exhaustive. This is due, in part, to the convenience of drawing the metadata from the project information at the same location.

2. Comprehensive Perl Archive Network (CPAN)

The Comprehensive Perl Archive Network (CPAN) repository, a successful and highly active resource for Perl developers, hosts similar search capabilities to

SourceForge. Items in the repository are grouped by type or function, and are then browsable within those categories. A keyword search over metadata is also possible. The metadata for CPAN is less expansive, but possibly more focused than the SourceForge repository data. The CPAN site also includes customer reviews of items in the repository.

As described, these examples of current repositories do not support all of the types of searches we would like to enable. The following section describes in more detail a few of our ideas for new search capabilities.

C. Improved Search and Discovery Capabilities

In addition to typical types of searches (keywords, popularity rankings), we envision a graphical user interface that enables navigation of repository assets depending on the users' interests. This requires an interface that allows users to project their context on the search mechanisms. In other words, the users bring particular information needs and goals based on the problem they are trying to solve. The interface needs to have natural mechanisms to enable users to pose inquiries that fit readily with their views of the problem space. For example, users may seek particular functionality best obtained through a functional organization of the information in the repository. Or, users may seek particular artifacts best obtained through a document resource organization of the information. Or, users may seek information on certain testing methodologies that have been applied so that a work activity organization of the information would best apply. The challenge in designing the framework for the software repository is devising initial sets of such taxonomic descriptions of the assets while creating flexibility for future introduction of additional and diverse organizational views (profiles or templates) of the information as user needs and repository utility grow.

1. Fish Eye Graph

One example of the type of tool that will be supported by the framework is a fish-eye graph (Sarkar & Brown, 1993). This is a visualization tool that has not been used, to date, to aid in navigation of repository contents. Fish-eye graphs display

objects of interest to users, along with the relationships the objects have with other items, as shown in Figure 1. As the relationships interesting to users are explored, the graph highlights the item and brings it to the front of the display. Users can then weed out uninteresting items by removing from view the relationships that are not important. This type of search results in a single or small grouping of items that users have found interesting with supporting information available by mouse-click.

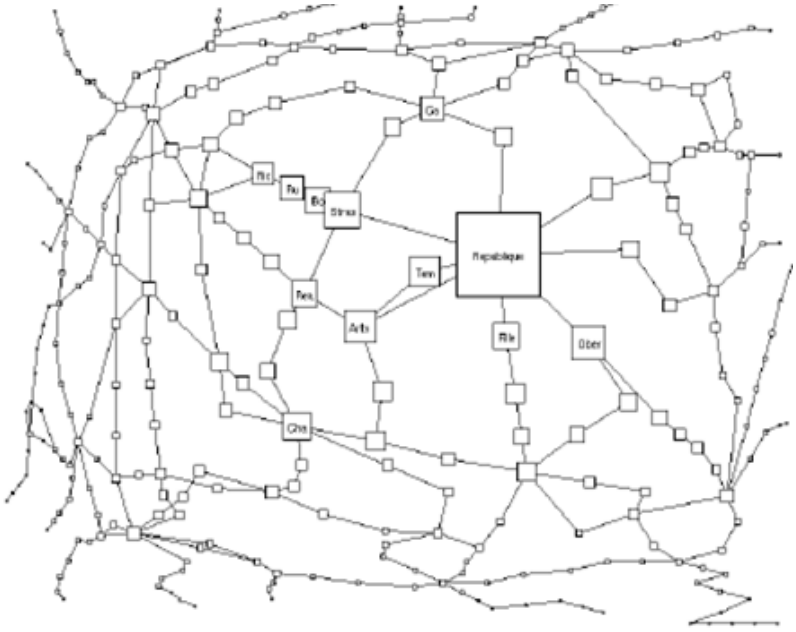

Figure 1. Example Fish-eye Graph (Sarkar & Brown, 1993)

2. Semantic Search

Current repository metadata schemas do not address issues of language ambiguity. Rather, they assume that keywords provided by the metadata will match identically to the words inserted by users. By providing a framework of related concepts in which to place the artifacts, a search tool can be designed to navigate for artifacts in such a way that the exact words initially used to describe artifacts need not be known.

A related ongoing NPS research project, titled ReSEARCH, focuses on solving these types of issues for SHARE. This work intends to enhance current

search mechanisms, principally Latent Semantic Indexing (LSI), by employing word-sense relationships provided in the extensive WordNet lexical database. However, this body of work lacks the domain-specific lexicon found in focused endeavors, such as Navy combat systems. Formalized semantic descriptions in the SHARE component specification and ontology will further enhance ReSEARCH capabilities to produce highly relevant search findings for users of the SHARE repository.

3. Model-based Search

A third type of search we have envisioned is based on a user-constructed model of the problem the user is trying to solve. The user interface for the repository can provide the capability to assist users in building the model of a desired system architecture using a standardized representation scheme (e.g., Unified Modeling Language), and the search can then return possible existing solutions for portions of the system and demonstrate where gaps likely exist. Model-based search has similarities to the semantic search concept described above—taxonomic and ontological descriptions of systems, system components, lifecycle phases, development artifacts, usage, and other concepts prominent in the software-hardware domain of SHARE provide structural information that can greatly facilitate search of available assets.

The metadata collected in current repositories do not support these types of advanced discovery tools. In the next section, we provide an overview of our approach for developing a repository that will enable these types of search capabilities.

III. Framework Overview

To enable the types of tools we envision, we must create a richer semantic framework for the repository. The framework will be composed of two parts—the component specification and the ontology.

A. Component Specification

The component specification is a description or model of the items in the repository. For our efforts we focus on two aspects of the component specification: the "typical" metadata and software behavior.

1. Metadata

The metadata for each artifact should incorporate all necessary data for discovery and implementation. The metadata will both aid repository users in determining if the item is suited for their use and will provide information about how to use the asset when it is retrieved. We refer to this as "standard" or "typical" metadata since there are many existing examples of metadata that we can use to develop the metadata for SHARE. Some of these examples are described in section IV.

2. Software Behavior

The metadata for many current repositories, such as those described earlier, fail to capture a searchable representation of the functionality of the items outside of general categories of functionality (e.g., Archiving Compression Conversion, Control Flow Utilities, Graphics, Security) and text-based search of code descriptions. Unlike current practice, the SHARE component specification will consist of both typical metadata and a behavioral model of the component. Since this piece of the component specification is not commonly incorporated into repositories in a standardized manner, we feel it is a specific focus area to identify the appropriate representation mechanisms for software behavior in the repository context. A discussion of relevant existing representations of software behavior is presented in section V.

B. Ontology

The second part of the framework includes descriptions of the relationships of the components, which form a contextual model of the repository items representing a particular perspective that can more closely match a user's problem context. These relationships may include the component's use/role in existing systems, its mapping to reference or domain architectures, its utility in various software development lifecycle phases, and other types of relationships we expect to discover during the research. Consider the example relationships among artifacts shown in Figure 2. Suppose we are inserting a requirements document for a particular component into the repository. This artifact may have been originally developed for System A in the figure. The item's relation to the rest of the original system provides the context for one dimension of the repository framework. If this item was then reused to fulfill some requirements of System B, its location in that model provides a second dimension. Additionally, the requirements document will map to some taxonomy of artifacts that are relevant for particular phases of the product lifecycle. Finally, the component it describes may also have a place in some domain-specific reference architecture. All of these relationships provide contextual information about the artifact that can be exploited to enable sophisticated search and discovery methods described in section II.C above.

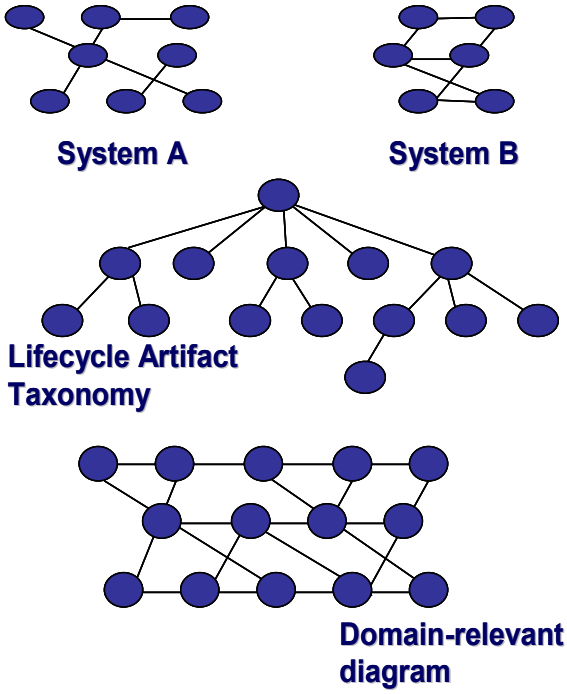

Figure 2. Artifact Relationships

For this project, an appropriate representation of component context will be identified and the relationships defined. This will enable navigation of the repository based on the contextual information provided in the ontology. Potential technologies considered for use in the ontology representation are discussed in section VI.

C. SHARE Framework Approach

Based on this vision then, the project team has identified three focus areas for developing the framework for the SHARE repository:

1. "Typical" metadata for artifacts
2. A suitable representation of software behavior
3. Framework relationships (ontology)

The current research project will focus on building each of these items for the SHARE repository. Follow-on work will be required to implement the framework in a tool suite that will enable the search capabilities described above. This and other suggested follow-on work is described in the Future Work section of this report.

In the next three sections, we investigate current initiatives and technologies that can be used in the development of each of these focus areas and evaluate the applicability for SHARE. Based on this assessment, we then describe our intended approach for developing each of these items in section VII.

IV. Metadata Initiatives

A. Introduction

Many researchers and developers are working on specification of metadata to describe assets and resources in various repositories. For the SHARE framework, we do not expect to create any unique approaches to developing metadata or that we will develop any fundamentally different metadata set. However, we intend to use the metadata descriptions to support navigation-by-context search, in addition to being able to do more traditional types of searches based on keywords, text-analysis, and popularity.

In this section, we discuss various initiatives that will guide the SHARE metadata development.

B. Web-based Technologies

The World Wide Web has experienced unprecedented growth over the past 20 years, fueled largely by the use of Hypertext Markup Language (HTML), Hypertext Transfer Protocol (HTTP), and Universal Resource Identifiers (URIs) as simplistic mechanisms for putting information into document files, posting and accessing those files, and linking those files, respectively. However, HTML primarily described how the information should be displayed in browser software, rather than providing clear descriptions of the content contained in the document. To address this shortcoming, the World Wide Web Consortium (W3C) created the Extensible Markup Language (XML) as a standard way to create and apply markup to the content of Web documents to make the content more readily accessible by software. While initial application of XML made description of Web content much more precise, it largely described content in a structured, syntactic manner. As the demand for greater automation in accessing and processing Web content continued to rise, principal designers and researchers on the Web created a new vision, called the Semantic Web.

The Semantic Web is Tim Berners-Lee's vision of the World Wide Web (Berners-Lee *et al.*, 2001) in which the vast stores of information become meaningful to computers and where "the explicit representation of the semantics underlying data, programs, pages, and other Web resources will enable a knowledge-based Web that provides a qualitatively new level of service" (Daconta et al., 2003, p. xxi). The Semantic Web is an extension of the World Wide Web in which information is given semantically-rich descriptions that enable automated processing by software. The W3C has created additional layers of markup, building on the base of XML, to provide description of the semantics of the information. The Semantic Web is an evolution of the current Web, built from the foundation of open standards on which the Web is built. Building blocks of the Semantic Web are shown in Figure 3. Below, we provide a brief description of the base layers of the Semantic Web stack (URI/IRI and XML) and highlight their relevance to the SHARE metadata development. In later sections of this report, we briefly describe the other components of the Semantic Web stack and discuss their relevance to the other parts in the design of the SHARE repository framework.

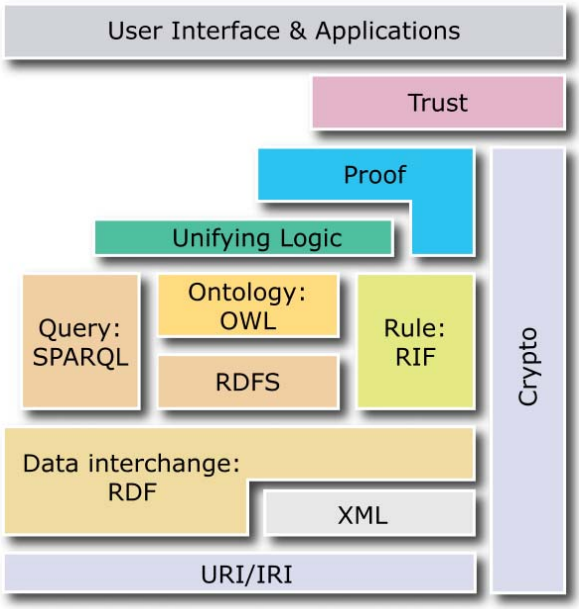

Figure 3. Principal building blocks of the Semantic Web Stack (W3C, 1994)

1. Uniform Resource Identifier / International Resource Identifier (URI/IRI)

The URI/IRI is an identification scheme for resources on the Web. The most common form is the Universal Resource Locator (URL) (a form of URI) generally used for links to documents in the HTML. Metadata records and library materials stored in the SHARE repository will likely have URIs assigned to facilitate discovery and access of those files using Web-based practices. We will also see in later discussion the use of URIs to identify abstract resources in the expression of assertions and relationships.

2. Extensible Markup Language (XML)

As introduced earlier, XML is a standard for defining markup languages. Markup languages enable information content to be self-describing for human and machine processing. The XML Schema language provides a capability to define the structure and content of XML documents that can be validated against the schema definition. For broadest utility, aspects of the SHARE component specification will be expressed in XML, beginning with development of an XML Schema to formalize the current set of metadata recorded to describe each asset in the repository.

C. Data Sharing Policies in the US Government

Recent policy decisions are driving significant efforts to revolutionize data sharing across the US government. Many of these are the result of presidential directives addressing protection of critical infrastructure and the ability to share information across agencies in times of national disaster. Various agencies may desire information on certain capabilities possessed by the military (and National Guard) for application in times of need (e.g., rescue and relief operations in national disasters) or in support of civilian crime prevention (e.g., counter-drug operations and counter-terrorism operations). Useful information may relate to platforms, air lift capacity, intelligence, surveillance, and reconnaissance capabilities, some of which may be available in the SHARE repository.

In the US Department of Defense, including DoD intelligence agencies and functions, the guiding document for information sharing is the Net-Centric Data

Sharing Strategy (DoD Chief Information Officer, 2003). The document defines net-centricity as “the realization of a networked environment, including infrastructure, systems, processes, and people, that enables a completely different approach to warfighting and business operations” (DoD Chief Information Officer, 2003, p. 1). The network foundation is the Global Information Grid, “the globally interconnected, end-to-end set of information capabilities, associated processes, and personnel for collecting, processing, storing, disseminating, and managing information on demand to warfighters, defense policymakers, and support personnel” (DoD Chief Information Officer, 2003, p. 1). Data assets addressed by the strategy include system files, databases, documents, official electronic records, images, audio files, web sites, and data access services. Users and applications can search for and “pull” data as needed, or they can receive alerts when data to which they have subscribed is updated or changed (publish/subscribe). The goals of the strategy are to make data (DoD Chief Information Officer, 2003, p. 10):

- visible—users and applications can discovery the data assets
- accessible—users and applications can obtain the data assets
- institutionalized—data approaches are incorporated into DoD process and practices
- understandable—users and applications can comprehend the data, both structurally and semantically, to address specific needs
- trusted—users and applications can determine the authority of the source of the data assets
- interoperable—metadata is available to allow mediation or translation of data to support many-to-many exchanges of data
- responsive to user needs—mechanisms for improvement through continual feedback are supported to address particular perspectives of data users

The design of the repository framework should provide or support mechanisms that address each of these goals. In this respect, the data sharing goals help to scope and guide the design and development efforts.

The data sharing strategy is being addressed through (1) self-organized Communities of Interest (COIs) for identification and maintenance of data; (2) metadata describing the data assets; and (3) GIG Enterprise Services supporting data tagging, sharing, searching, and retrieval. In the Department of the Navy, numerous COIs have formed in recent years, including Anti-Submarine Warfare (ASW), Undersea Warfare XML (usw-xml), Maritime Domain Awareness (MDA), Mine Warfare, and Service Oriented Architecture (SOA) Transformation Group. Navy representatives play a strong role in the DoD Modeling and Simulation (M&S COI), among others. There is a proposed COI for Software Asset Management being organized by the Defense Information Systems Agency (DISA) that may be pertinent to SHARE development as well.

In the data sharing strategy, data assets are described by metadata to support discovery by users and applications. A standard set of metadata for discovering distributed resources is provided in the DoD Discovery Metadata Specification (DDMS) (Deputy Assistant Secretary of Defense, 2007). The DDMS states:

> Data assets available on the Enterprise must be described with metadata, using the information elements defined in this document to permit discovery through the Enterprise Discovery capability. The DDMS defines a core set of elements that must be used to describe assets made visible to the Enterprise. Users (human and systems) that search the Enterprise will discover data assets that have been tagged and entered into catalogs or repositories that respond to search queries specified in terms of DDMS entries. (Deputy Assistant Secretary of Defense, 2007, pp. 16-17)

The SHARE repository, as with other repository efforts, can readily address this directive by ensuring that sufficient metadata are provided in descriptions of assets to allow generation of at least the minimum required set of metadata specified in the DDMS.

The concept for use of DDMS for asset discovery is shown in Figure 4.

The requirement to support DDMS does not preclude using more sophisticated and domain-centric metadata to describe assets in the SHARE repository. Nor does the DDMS standard preclude development of more sophisticated search mechanisms for repositories like SHARE. It simply defines a minimum level of standardized metadata that will be supported by GIG Enterprise Services. In fact, the DDMS design reflects a combination of a core layer of metadata with an extensible layer providing COI/domain-specific metadata, as shown in Figure 5. The Summary Content Category Set of the DDMS is specifically aimed at providing "content-related" details about data assets. Content metadata provides topics, keywords, context, and other content-related information; gives users and applications insight into the meaning and context of the data; and provides a basis for search engines to perform searches for data assets that address specific topics (DoD Chief Information Officer, 2003, p. 15).

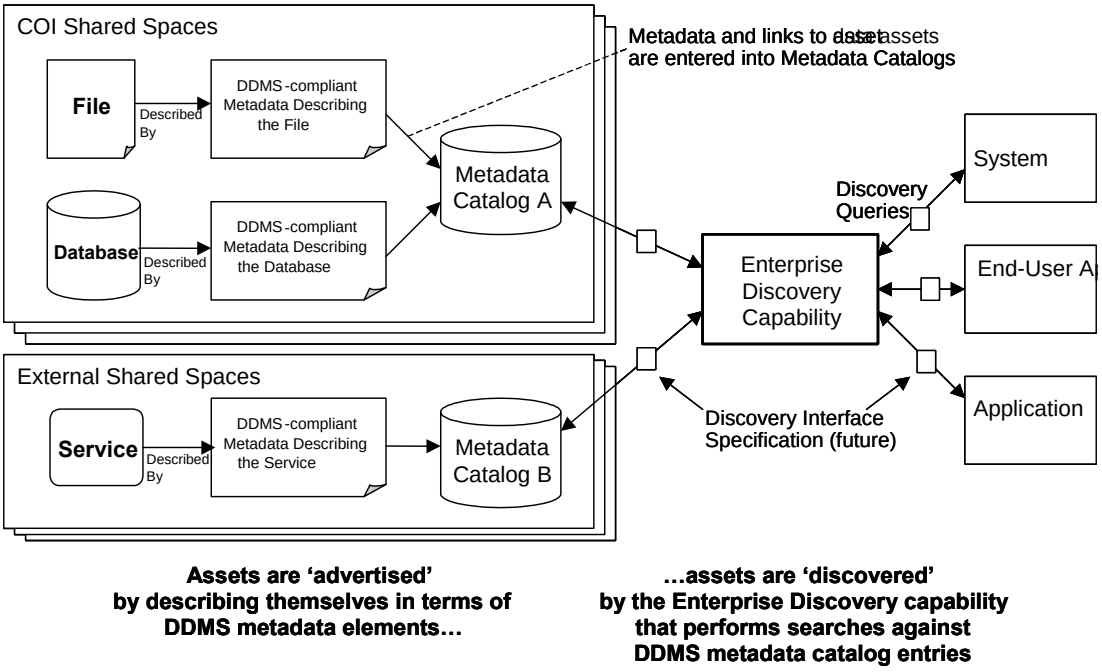

Figure 4. DDMS Usage Concept (Deputy Assistant Secretary of Defense, 2007, p. 19)

The DoD Modeling and Simulation (M&S) COI is actively defining metadata for discovery of assets. This group recently published an initial DoD M&S Discovery Metadata Specification (SimVentions, 2008) built upon the information requirements

of the DDMS. It is interesting to note that M&S resources will likely become an important subset of the SHARE repository. Increasingly, system development includes application of modeling and simulation for proof of principle analyses, testing, and training. Policies have promoted Simulation Based Acquisition (SBA) for many years. These resources represent significant investments that can potentially be reused in support of future systems. In order to ensure compatibility of SHARE with the M&S community and DDMS metadata initiatives, we will ensure that the SHARE component specification framework also supports generation of discovery metadata sufficient to meet their broader requirements.

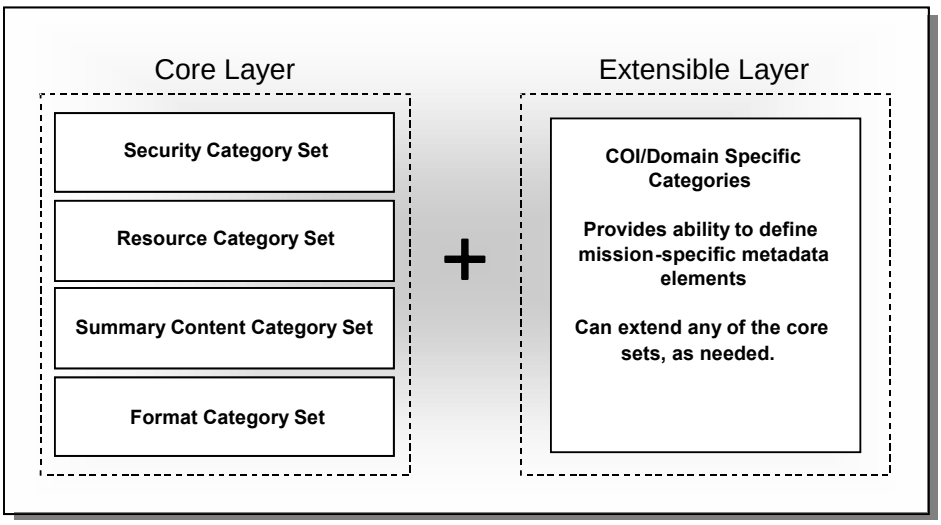

Figure 5. DDMS Logical Model Consisting of a Core Layer and a COI/Domain Specific Extensible Layer (Defense Assistant Secretary of Defense, 2007, p. 21)

The GIG will provide a number of core enterprise services, including Discovery, Messaging, User Assistant, Information Assurance / Security, Enterprise Service Management, Storage, Mediation, Collaboration, and Application. As these GIG services become available, it will be advantageous to adapt the SHARE architecture to employ these services. While using the common infrastructure, this will also open the SHARE content to the broad DoD community through the standardized practices.

GIG Enterprise Services also include the DoD Metadata Registry (MDR). This registry, based on the International Organization for Standardization (ISO) 11179 specification for metadata registries, is available throughout the Enterprise. The Registry represents a “one-stop shop” for developer data needs and is a key component in achieving the Department’s interoperability goals:

All document formats, interface definitions, and exchange models used by systems will be stored in the DoD Metadata Registry. Developers can discover these metadata assets and utilize them to read, write, or exchange data that is made available throughout the Enterprise. All programs and COIs have a responsibility to support interoperability through active participation in the DoD Metadata Registry. The DoD Metadata Registry will provide capabilities to further support interoperability through the use of translation and mediation services and for the sharing and reuse of processes.” (DoD CIO, 2003, p. 8)

The Net-Centric Data Sharing Strategy directs COIs to take the lead in establishing COI-specific metadata structures, defining community ontologies, cataloging data and metadata, and having members post data. A *community ontology* “provides the data categorization, thesaurus, key words, and/or taxonomy” that can be used to “increase semantic understanding and interoperability of the community data” (DoD Chief Information Officer, 2003, pp. 5-6). Taxonomies “enhance discovery by providing a hierarchical means of searching for data while providing users and applications with additional insights about data assets by indicating their placement among other data assets” (DoD Chief Information Officer, 2003, p. 15). Furthermore, COI-developed vocabularies will define terms used in describing data assets, and the thesauruses will identify related terms to assist translation services. As we will see, the defined vocabularies, taxonomies, and ontologies will serve an important role in enhanced asset search and discovery in the SHARE repository. We anticipate posting schemas, taxonomies, and ontologies developed for the SHARE repository framework to the MDR to support community information sharing and data mediation.

D. Commercial Metadata Practices

Outside of the DoD, there are many additional metadata practices from which we can learn. All existing repositories have some sort of metadata schema, whether well defined or not. Also, there are some specific efforts focused on the development of metadata standards for use in software repositories. Some examples of these are discussed here.

1. Existing Repository Metadata

As previously discussed, open source repositories such as SourceForge and CPAN have a metadata set for describing the assets they contain. Unfortunately these schemas are not often published. However, they can be somewhat derived by looking at the available information for each of the items in the repository. As an example, see the SourceForge and CPAN metadata sets derived in Figure 6.

For SHARE metadata development, these existing examples of metadata will be used as a reference when the metadata schema is developed. Existing metadata sets will be used to trigger the evaluation of items that could be included that were not originally considered. The goal is not to merge all existing sets of metadata but to assess the relevance of existing data sets for SHARE and include any appropriate items.

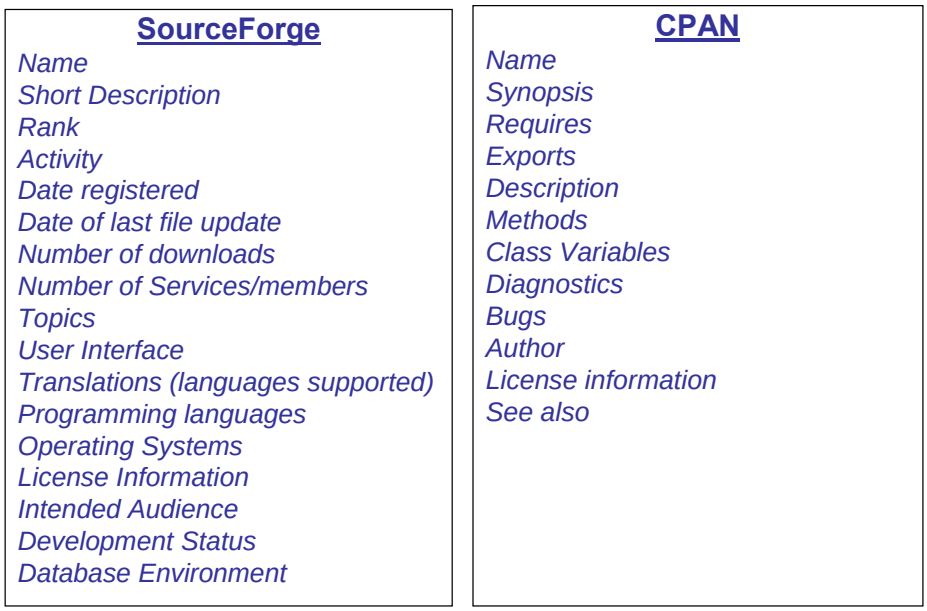

Figure 6. Example Metadata from SourceForge and CPAN

2. Object Management Group Reusable Asset Specification

The Object Management Group (OMG) created the Reusable Asset Specification (RAS) to standardize the packaging of software assets. The RAS describes required and optional classes, as well as required and optional attributes, for packaging software assets. The specification is depicted as Universal Modeling Language (UML) models which are translated into XML Schema and Meta-Object Facility (MOF) / Extensible Metadata Interchange (XMI) XML Schema.

In the RAS, artifacts are defined as "any work products from the software development lifecycle," and assets are a grouping of artifacts which "provide a solution to a problem for a given context" (Object Management Group, 2005, p. 7). Accordingly, the RAS describes an approach for packaging *artifacts* into an *asset* using a manifest file, also an XML document.

It is worth noting that these definitions of artifact and asset are similar to those definitions adopted by the Navy Open Architecture (OA) program. However, while the RAS focuses on asset discovery, we believe it is desirable to enable discovery of these pre-packaged assets as well as user-defined assets. After all, when inserting items into a software repository it is not likely that every desired configuration of

artifacts into assets can be determined ahead of time. Rather, it is more likely they would be determined at search time. This is because the ability of a group of artifacts to "provide a solution to a problem" will depend on the needs of the searcher. For example, a requirements document (an artifact) could be incorporated into several different types of assets depending on whether the problem at hand pertains to developing requirements documents for similar systems or to understand a particular system completely. If the former, the desired asset may be a package of similar requirements documents. If the latter, the desired asset may be the complete set of available artifacts associated with a particular system, including requirements documents, code, test cases, etc.

It is therefore our intention to apply metadata descriptions to artifacts, so that assets can be determined by the searchers needs. This does not preclude the pre-packaging of artifacts into assets to solve common problems. We envision the capability for searchers to discover a problem solution by either locating a prepackaged asset, or by building an asset from artifacts they believe are necessary.

Because of this distinction, there are significant portions of the RAS that will not be relevant in developing the appropriate data schema for SHARE. Additionally, many of the RAS attributes are only applicable for certain types of assets. We propose to create metadata that can be tailored to the different artifact types. This will result in one core set of metadata for all artifacts and additional fields required depending on the type of artifact being described.

E. Summary

Several patterns of use of metadata for describing software assets are prevalent in government and industry. Starting with the current descriptions in the SHARE repository, we will apply best practices to formalize descriptions to create the foundation for the repository framework.

THIS PAGE INTENTIONALLY LEFT BLANK

V. Software Behavior Representation

A. Introduction

Repositories today tend to capture software behavior as key words describing a general functional area or as a free text description field in the metadata. This type of description can be helpful for aiding users to determine if the item will be useful in meeting their needs. However, if the desired end goal is more sophisticated than today's repository capabilities, a more formal description of behavior is required. For example, one of the loftier goals of a software repository may be to automatically compose systems from reusable components. This is a difficult problem, which many have tried to solve[3]. It is especially difficult if the components were not originally designed for reuse. As a necessary first step towards more sophisticated uses of a repository, behavioral descriptions must be machine-readable in order to support automated search and discovery. Furthermore, the behavior descriptions must be formalized and consistently applied to each item in the repository if the intent is to automatically compose them into a larger functioning system.

In this section, we discuss the various methods currently used to formally express software behavior. Each of these types of representations has advantages for certain purposes and may or may not be suitable for use in SHARE. As we discuss the various types of representations, we include our initial assessment of whether we should implement the method in SHARE. A key consideration in this determination is the level of effort required to produce the descriptions. The wide array of contributors to SHARE requires caution in dictating standards that will impact the development processes of the asset developers. Therefore, in our

[3] The proceedings from the International Symposium on Software Composition, an annual event, provide examples of research into the breadth of research topics currently being pursued in the area of software composition. The web site for the 2008 conference is located at http://www.2008.software-composition.org/

assessment of the approaches below, we seek a balance between method robustness and ease of implementation.

A formalized description of software behavior typically means one of two things. We either (1) define the inputs and outputs (interfaces) of the components, or we (2) describe the operations that take place within the component. Many people view the latter as a decomposition of the former. In other words, they describe the inner workings of a component by defining the inputs and outputs of a more granular subset of components. Therefore, we have summarized the current approaches for documenting both software interfaces as well as software behavior.

B. Interface Descriptions

Interface descriptions focus on the inputs and outputs of a component and not the inner workings of that component. Interfaces are represented using various methods, which vary from concentration specifically on the connect points between two pieces of software and the types of information passed between them, to representations of the services that a component provides.

1. Interfaces as contracts

One method for representing interfaces often employed in component software technologies is as a contract between the client and the provider of the implementation (Szyperski, 2002, p. 53). The contract defines the services promised by the interface and the requirements of the client for using the interface. It could simply consist of a set of named operations that can be invoked by clients. It may also include pre- and post-conditions necessary for the successful use of the interface.

A drawback for using this type of interface description as a basis for search and discovery in SHARE is the dependency on the component's originating software language for determining the syntax and semantics used to describe the operations and conditions. In SHARE's heterogeneous environment, these types of standardized descriptions may not be practical.

2. Interface Definition Languages (IDLs)

Component technology developers have developed Interface Definition Languages (IDLs) to specify interfaces independently of the programming language used for source code development (Clements *et al.*, 2003, p. 241). Examples include OMG IDL and Microsoft's COM IDL, which are demonstrated in Figure 7. and Figure 9. , respectively.

```
interface salestax  {
float  calculate_tax ( in float taxable_amount );
}
```

Figure 7. Sample OMG IDL Interface. (Object Management Group, 2007)

Figure 8.

```
[object, uuid(348ACF20-C9B9-11d1-ABE5-966A46661731)]
interface IDerivedInterface : IUnknown
{
import "unknwn.idl";
import "wtypes.idl";
HRESULT Fx(int iValue);
}
```

Figure 9. Example COM IDL Interface. (Hludzinski, 1998)

The same drawback discussed for the programming language-dependent contracts for our heterogeneous SHARE environment exists for these intermediate languages. Rather than a dependency on the programming language, however, the dependence lies in the chosen component technology. Since we do not intend to force a specific component technology for all SHARE contributors, it does not make sense to insist on interface definitions based on these IDLs.

3. Architecture Description Languages (ADLs)

Primarily used to formally represent system architectures for use during development, Architecture Description Languages (ADLs) typically describe system elements, their interactions and their composition rules. While there are many different viewpoints about what constitutes an ADL (Medvidovic & Taylor, 2000, pp.

71-72), they always include a formal description of interfaces. ADL interface descriptions typically define the required and provided services (messages, operations, and variables) of a component. Some ADLs also allow for parameterization of interfaces. Others provide additional information. Rapide uses partially ordered sets of events, or posets, to describe behavior and component interaction (Clements, 1996, p. 21), while MetaH explicitly formalizes the algorithms performed within the component in a domain-specific language (Medvidovic & Taylor, 2000, p. 79).

The advantage of using ADLs in a component specification is that the benefits of ADL-based tools may be realized for the components. ADL tools assist the developer by supporting architecture creation, visualization, validation, refinement, simulation, and analysis, in addition to features that enable systematic transformation of architectures into the implementation of a system. Many support generation of “glue code” for components once their implementations are developed. Additionally, ADLs are likely the appropriate level of abstraction for a heterogeneous collection of assets, such as that found in SHARE, since they do not depend on any decisions made about the implementation of the components.

Unfortunately the use of ADL-type descriptions comes with a cost. Because of the robust descriptive capabilities of many ADLs, there is considerable effort required in learning to use them. This would present a learning curve for both asset submitters and retrievers. To minimize this problem, tools could be developed to aid users in producing the required ADL descriptions. As an alternate solution, we are investigating the possibility of incorporating some ADL-like descriptions into the XML-defined metadata for the components. This will enable us to incorporate only relevant aspects to the SHARE repository. Several new XML-based ADLs such as XML-based Architecture Description Language (XADL) (Zhang, Ding, and Li, 2001, pp. 561-566) and Service Oriented Architecture Description Language (SOADL) (Jia *et al.*, 2007, pp. 96-103) may form the basis for this development.

4. Graphical Notations

Interfaces can also be represented using UML or other graphical notations. Typical graphical notations of interfaces include the “lollipop” depiction or the expression of an interface as a UML stereotype. These are demonstrated in Figure 10.

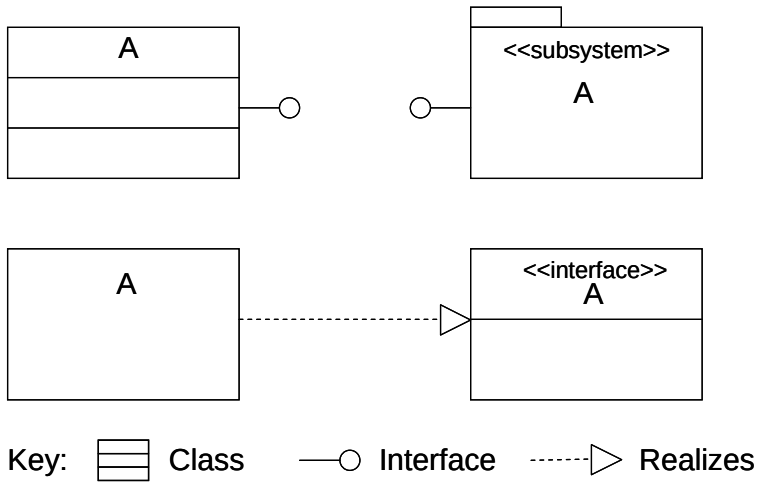

Figure 10. UML Interfaces
(Adapted from Bass *et al*., p. 219)

Often these pictorial depictions of interfaces are further defined using a formalized language such as the OMG IDL described earlier (Clements *et al.*, 2003, p. 241). In addition to the visual aid provided by the diagrams, the value of using UML for interface descriptions is that many tools have been developed to read UML and translate the models into XML depictions (XMI) and into executable code. Model Driven Architecture products are available that enable the automatic development of “glue code” between components from the architecture specification (Frankel, 2003).

On the downside, an object-oriented programming development paradigm is assumed. While some generality can by achieved by using packages and subsystems as the main UML building blocks instead of classes and subclasses, some argue that attempting to use UML outside of the arena for which it was designed is more trouble than it is worth (Shaw & Clements, 2006, p. 34). This realization, as well as our understanding that whichever description method is

chosen must be applied across multiple development cultures, compels us to assert that UML may not be the best way to represent interfaces for SHARE.

5. Service Oriented Architecture and Web Services

Future deployment of the SHARE repository is likely to evolve toward the Service-Oriented Architecture (SOA) of the GIG. SOA has been described as “an ideal vision of a world in which resources are cleanly partitioned and consistently represented” (Erl, 2005, p. 3) and “automation logic is decomposed into smaller distinct units of logic … known as *services*” (Erl, 2005, pp. 23-33). Elements of a service architecture are similar to SHARE concerns—the architecture typically includes a registry of services containing descriptions of those services and information on how to access them. Mechanisms are provided for service discovery and for passing sufficient information about the service back to the caller so that the service can be invoked. Advanced concepts include service orchestration for composing higher order services from component services. The focus, of course, is service reuse, which will potentially reduce development and maintenance while improving software reliability and evolution agility. Figure 11. identifies a number of service-orientation principles related to service reusability.

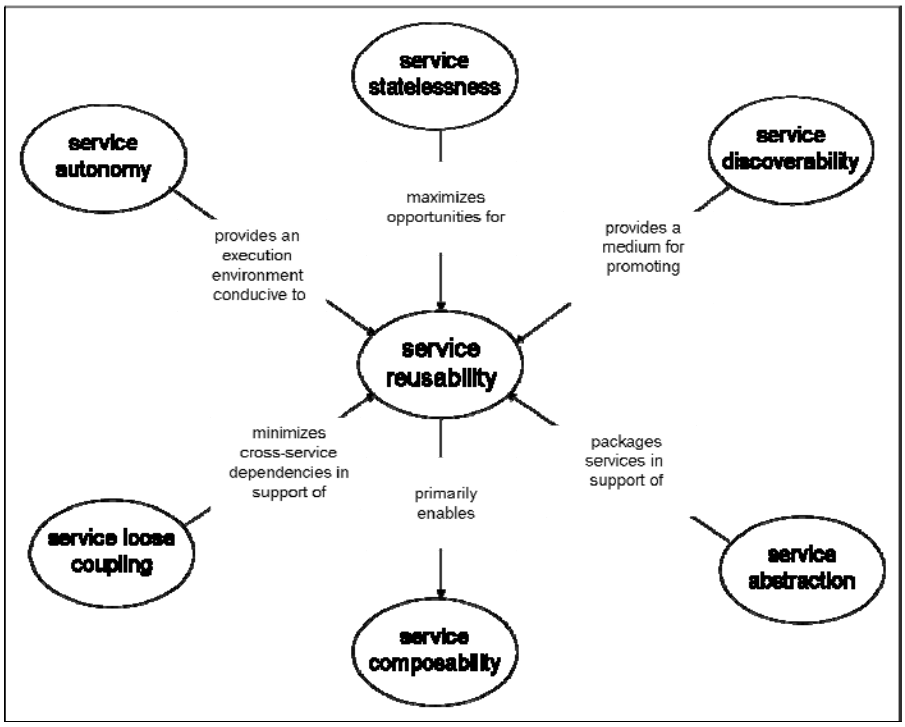

Figure 11. Service reuse and relation to service-orientation principles (Erl, 2005, p. 313)

SOA realization may employ Web Services standards such as: Universal Description, Discovery, and Integration (UDDI)[4] for creating service registries; Web Services Description Language (WSDL)[5] for identifying operations offered by services and describing input/output interfaces for those operations; the Simple Object Access Protocol (SOAP)[6] for accessing services and passing data to/from the services; Web Services Business Process Execution Language (WS-BPEL)[7] for describing workflow logic for orchestration of services; OWL for Services (OWL-S)[8], an ontology of services supporting service advertisement and discovery, description of service operation, service interoperation; Web Services Interoperability (WS-I) profiles[9] describing collections of Web services specifications at specific version

[4] For UDDI information, see: http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm
[5] For WSDL information, see: http://www.w3.org/2002/ws/desc/
[6] For SOAP information, see: http://www.w3.org/TR/soap/
[7] For WS-BPEL information see: http://www.ibm.com/developerworks/library/specification/ws-bpel/
[8] For OWL-S information, see: http://www.w3.org/Submission/OWL-S/
[9] For WS-I information, see: http://www.ws-i.org/

levels; and others. It is interesting to note that the problem of describing Web services in sufficient semantic detail to enable automatic composition of services is very similar to the problem as describing software components for reuse.

In Web Service implementations, XML is generally used to hold the information passed across an interface. XML schemas are extensible and easily modified if there is a need to change the standardized format of the data. The above standards for describing and implementing Web Services are XML-based specifications. Additionally, XML is readily digestible by many existing tools and is well enough understood universally to be implemented into new ones. These advantages motivate us to propose XML as the primary notation for documenting metadata, including the interfaces, for the SHARE component specification and ontology project. The flexibility of XML will enable us to incorporate the necessary information to enable capabilities similar to those enabled by ADLs, without the high overhead cost of training the end users. Although SOA and Web Services are in a high state of flux as industry standards mature, they present opportunity to create software component specifications in SHARE that can be employed for a number of purposes.

C. Modeling Software Behavior

In addition to understanding the interfaces for a component, a repository user is interested in the functionality of the software components. In this section, we discuss the notations currently used to describe the activities that take place within a component.

1. UML

In addition to the structural diagramming capabilities provided by the Unified Modeling Language (UML), several types of diagrams are used to model dynamic aspects of the system. Methods for formal documentation of behavior provided by UML include sequence diagrams, which may be further amplified using a constraint language such as UML's Object Constraint Language (OCL), collaboration diagrams, and statecharts.

Sequence diagrams, or message sequence charts, show the interactions of objects within a component in a time-ordered sequence (Larman, 2005, pp. 222-225), as shown in the simple banking example in Figure 12. The boxes at the top of the diagram are the objects, and the messages that take place between them are ordered sequentially from top to bottom. Collaboration, or communication, diagrams also show objects and their interactions, but in a more condensed format that tends to lose the visibility of the time-ordered sequencing (see Figure 13.

State Machine Diagrams, or statecharts, illustrate events and states of objects (Larman, 2005, pp. 485-492). As shown in Figure 14. states are represented by the rounded rectangles, and the possible state transitions are indicated by the arrows connecting them. Amplifying information such as actions triggered by transitions and activities that take place during particular state conditions is often included.

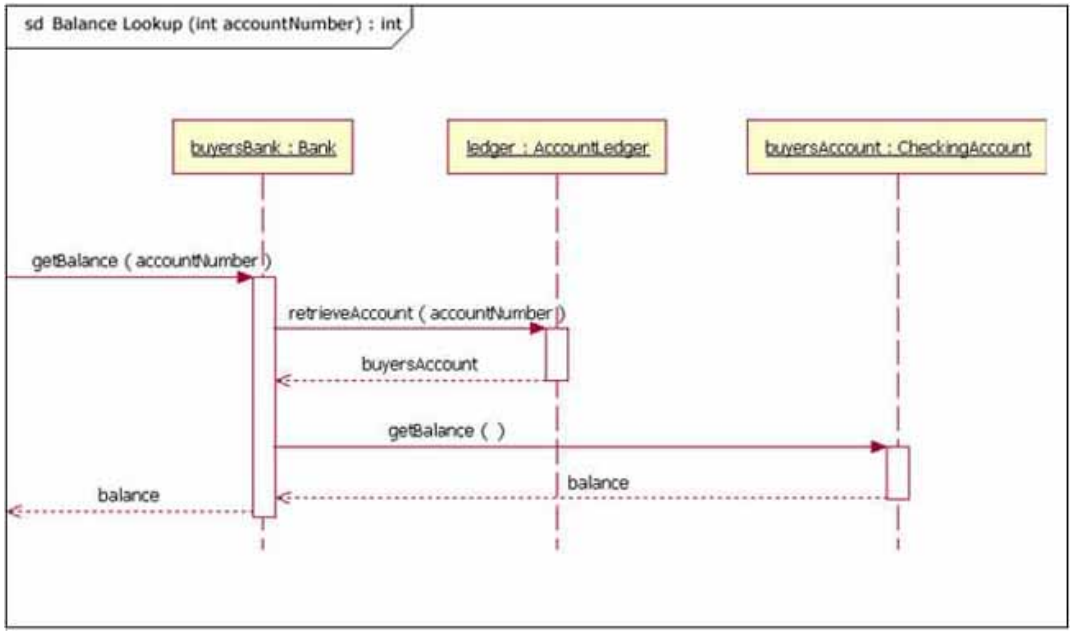

Figure 12. UML Sequence Diagram (Bell, 2004)

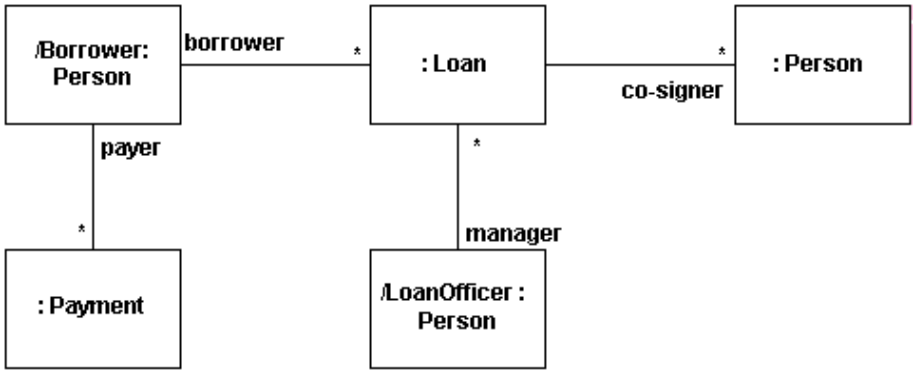

Figure 13. UML Collaboration Diagram (Ambler, 2007)

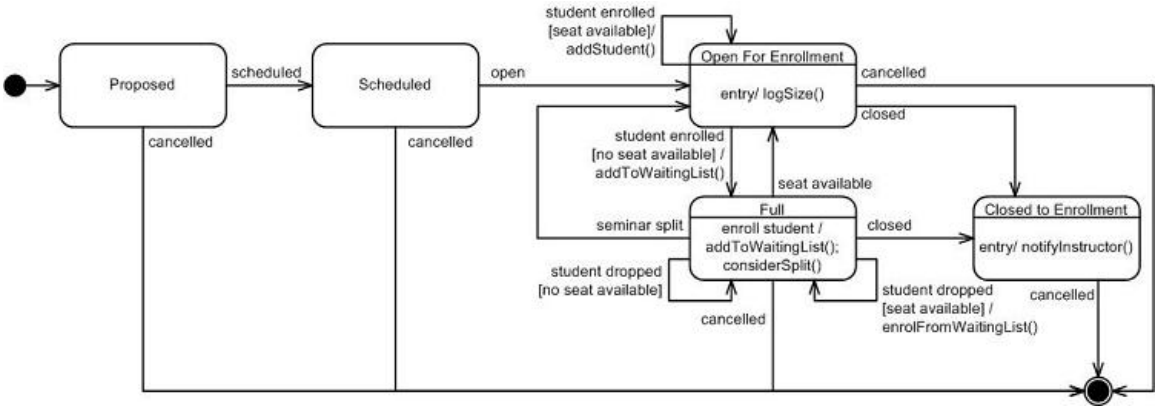

Figure 14. UML State Machine Diagram (Ambler, 2006)

Each of these UML diagrams sheds some light on particular aspects of a component's behavior and could be used to formalize the behavioral descriptions of artifacts incorporated into SHARE. There are a few drawbacks to this approach, however. First, as discussed previously, the use of UML diagrams often assumes an object oriented development paradigm, which may not be relevant for all SHARE submitters. Second, the UML tools presented primarily assist in system development and may not be best suited for asset discovery and retrieval. Repository users are likely to be more interested in a more abstract view of the system than this implementation level information provides. Finally, each of the diagrams only captures a particular "slice," or view, of the software's behavior. For a

complete behavioral description, it would be necessary to require each type of diagram plus additional information. This would result in a steep overhead to develop this information for each item contained in SHARE. For these reasons, we do not anticipate incorporating UML activity/state diagrams as the standard representation method for software behavior. However, if these depictions are generated as part of the software engineering development process, they should be included as artifacts in the repository.

2. Formal Languages

In formal specification, system behavior is described using mathematical structures. Formal notations that enable this type of specification include the Vienna Development Method (VDM), Z (pronounced *zed*), and Alloy. Since the languages are mathematically based, developers can use logic to reason about a formally specified system and sometimes prove its correctness.

As a small example, consider Spivey's basic birthday example shown in Figure 15. (Spivey, 1992, pp. 3-7). The schema defines a state space for a system that records people's names and birthdays. The portion above the line declares the variables *known* and *birthday*, and the portion below the dividing line provides the relationship between variable values. *Known* is the set of names for which there is a recorded birthday, and *birthday* provides the date of the associated birthday. The invariant provided below the dividing line states that the set *known* is equal to the domain of the partial function *birthday*.

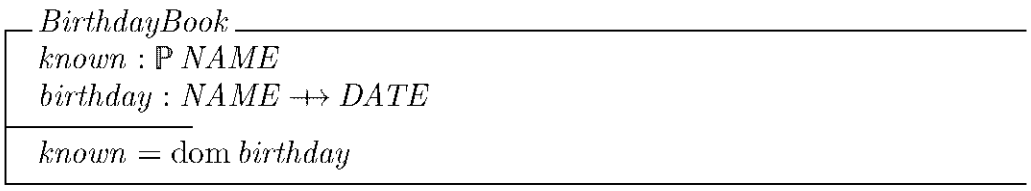

Figure 15. Z State Space Notation (Spivey, 1992)

To specify an operation that takes place in a system, the relationships of the variables before and after the change are described in the bottom portion of the operation schema. Before values are listed as the variable name (*birthday*), and after values are listed with the apostrophe symbol (*birthday'*). In the operation depicted in Figure 16. the operation adds a name/date pair to the previous set of all birthdays.

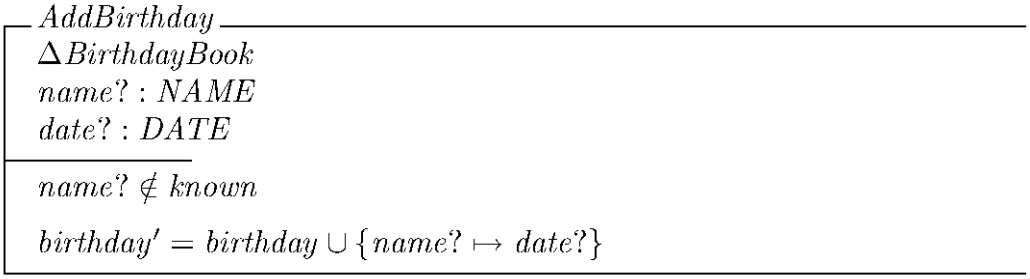

Figure 16. Z Operation Notation (Spivey, 1992)

It is evident that with even this small example, a solid understanding of set theory, logic and other mathematical foundations are desired when learning how to construct specifications in Z. This is one of the complaints about formal languages, as well as one of the reasons that the use of formal specification is mostly a topic of research and limited in practical applications to systems or portions of systems with safety-critical reliability demands.

MIT's Alloy project is one of the more successful attempts to make formal methods more user-friendly (Jackson, 2006). Alloy helps the user develop the specification by providing a visual simulation of the model. This enables users to

recognize when the model is incorrect, and they can then iteratively develop the model in more detail. Alloy also includes an analyzer that automatically checks invariants for inconsistencies in the model.

Even with these advances, however, the amount of effort required to specify systems in these formal notations is well above the desired level of effort threshold for the SHARE repository. Therefore, we do not intend to use formal languages to represent software behavior of assets in SHARE.

D. Summary

For SHARE, we do not hope to solve the composition problem in the near term. Mandating formal descriptions of software behavior for repository items does not seem worthwhile when the composition problem remains unsolved. However, intermediate steps towards formalized behavior descriptions will prove useful in the near term and helpful in advancing towards far-term goals. To this end, we are currently planning to extend the XML-defined metadata to incorporate interface information as well as existing reference architecture information to standardize behavioral descriptions for each artifact entered into the repository. Ongoing advances in service composition in SOAs will also be examined for application to the framework.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. Relationships Framework (Ontology)

A. Introduction

Rich ontologies capturing the relationships of entities from multiple views have not been applied to software repositories. However, there are many examples of the use of ontology in the organization of data for different applications.

As an example, consider the intelligence community's challenge of synthesizing disparate pieces of information from widespread sources into logical connections to form coherent pieces of knowledge. There are currently several applications designed to collect the data and assist the analyst in drawing relationships between the data.

Palantir Technologies has created one such software application to support the DoD intelligence community by providing robust capabilities for managing data from various sources[10]. The Palantir tool is based on user-defined ontologies and supports multiple representation and analysis tools. The graphical representations depict the data items and their relationships with each other, based on the underlying ontology. The analysis tools can be used to form logical links between entities in the database and to detect patterns and irregularities in the data. This rich environment enables multiple search techniques including keywords, browsing through data tables, and graphical views of the database content based on the relationships of the entities described in the ontology (see Figure 17.).

For the SHARE research project, these capabilities serve as examples of potential utility of the repository framework by demonstrating the power of formalized semantics. When the framework is in place, technologies such as these can be exploited to gain flexibility in the search options described previously.

[10] For more information about Palintir products, see: http://www.palantirtech.com/

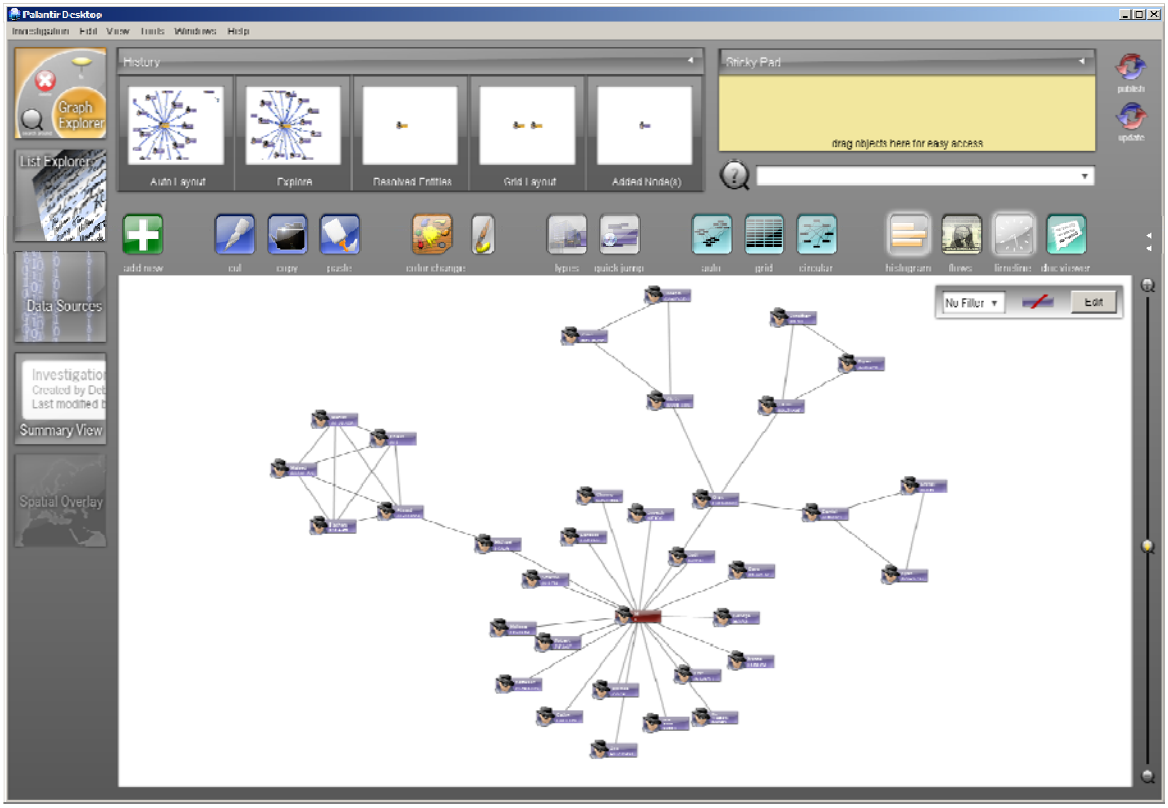

Figure 17. Palantir Graphical Interface (Gordon-Schlosberg, 2008)

Similar examples of the use of ontologies to support data analysis exist in other domains, particularly in the medical field. Some background on current and emerging standards for describing rich semantics in data relevant to the SHARE framework is provided in this section.

B. Semantic Web Techniques

The Semantic Web stack was shown in Figure 3. Several of the components pictured there contribute to stronger semantic description of Web-based resources, as described below.

1. Data Interchange: Resource Description Framework (RDF)

RDF is a language for stating assertions in the form of subject-predicate-object triplets. Each of the elements in an RDF statement is an abstract Web

resource identified by a URI. RDF and its schema language (RDFS, see below) will be investigated for applicability to the SHARE framework to describe taxonomies (class hierarchies) supporting inference and search (Alesso & Smith, 2006). We will also explore the possible benefits of creating RDF expressions for storing SHARE repository data content.

2. Query: SPARQL

The lower layers of the Semantic Web stack provide the ability to describe information (metadata and schemas) and to express knowledge (assertions). Query languages provide a means to access information. The XML Query language is used to search XML documents by exploiting the hierarchical tree structure of the documents (XPath expressions). The SPARQL Protocol and RDF Query Language provide a means to search RDF expressions by exploiting the subject-predicate-object graph structure of the expressions (pattern matching).[11] If RDF structures prove valuable for describing information in the SHARE repository, the use of SPARQL and other query techniques will be explored.

3. RDF Schema (RDFS)

RDF provides the means to make statements about Web resources. RDF Schema (RDFS) provides an XML vocabulary to define classes and subclass relationships (taxonomies) as well as to define properties associated with classes (ontologies) (Alesso & Smith, 2006). RDFS will be explored for use in description of taxonomies and, as we will see below, as part of the specification of ontologies for the SHARE framework using the Web Ontology Language (OWL).

4. Ontology: Web Ontology Language (OWL)

OWL extends RDF/RDFS constructs to provide more precise description of classes, subclasses, and relationships among classes (properties). OWL adds the capability to define local scope of properties, disjointness of classes, Boolean combinations of classes, cardinality restrictions, special characteristics of properties

[11] For more information on this language, see: http://www.w3.org/2001/sw/DataAccess/

(e.g., functional, transitive, symmetric), and other aspects not expressible with RDF/RDFS (Alesso & Smith, 2006). In addition to using the language to describe classes and relationships, OWL also describes instances (members) of classes, which allows creation of knowledge bases containing information about the software-hardware assets in SHARE. OWL includes three sublanguages (OWL Full, OWL DL, and OWL Lite), providing three levels of logical expressivity and resultant computational trade-offs. OWL Lite is the simplest of the three, excluding the ability to define enumerated classes, disjointness statements, and arbitrary cardinality (Alesso & Smith, 2006). OWL DL (Description Logic) permits expression of a subset of first order logic that guarantees decidability (determining an answer in finite time). If determined to be appropriate for our purposes, we will use OWL DL for ontology development for the SHARE framework. Use of OWL will maximize utility by software applications, including use of openly available reasoning engines that can be used to check for ontology consistency and to make inferences about instances in the asset knowledge base.

5. Rule: Rule Interchange Format (RIF)

Rules and rule-based systems provide additional expressiveness in describing the logic of a system. Rules permit software to infer a conclusion from a premise (Alesso & Smith, 2006). Rules may be used in the formalized specification of software assets in the repository to enrich their description, particularly if there is a need to encode business rules, policies, and processes appropriate to the repository (e.g., role-based access).

6. Unifying Logic and Proof

The use of the well-established Web-based conventions in the information technology community provides a basis for application of a variety of common logical computations. We will be able to employ existing products that can operate on the semantic descriptions using provably correct methods.

7. Cryptologic

Cryptologic aspects of the Semantic Web stack cut across all the layers, supporting such functionality as authentication, encryption, and digital signature

(Eastlake & Niles, 2003). We will not address this area directly in the work, but we will create the semantic basis for implementation of methods such as role-based access and other controls on information content in the repository.

8. Trust

Trust is being able to anticipate the actions of a system and have a reasonable expectation that the system will act correctly (i.e., as intended) (Michael, 2008). Trust is often established and maintained through transparency. One of the advantages of the use of the Semantic Web practices is visibility of the information through its description in metadata, semantic descriptions, rules, and computationally sound logic. Clearly, users of the repository will rely on the trustworthiness of the content when obtaining information or artifacts that support new developments. While we will not address this aspect of the problem directly, in the component specification and ontology development, our goal is to make the information as explicit and accessible as possible to humans and machines to promote this level of the Semantic Web stack.

9. User Interface & Applications

Well-defined syntax and semantics for description of metadata, taxonomies, and ontology for the SHARE framework will facilitate development of software applications and user interfaces for working with the repository. By expressing the SHARE component specification and ontology using common Semantic Web elements, the products of our current research will readily support development of various applications, including Web Services in an SOA, while also providing a basis for future applications employing emerging Semantic Web Services technologies.

C. Semantic Search

Semantic search methods “augment and improve traditional search results by using not just words, but meaningful concepts” (Alesso & Smith, 2006, p. 201). One prominent approach, Latent Semantic Indexing, considers documents that share many words in common to be semantically close, without any understanding of the “meaning” of the words. As introduced earlier in this report, other researchers at

NPS are developing semantic search capabilities (ReSEARCH) for the SHARE repository that will use the WordNet database to extend this approach to include related words (synonyms, part-of relationships, etc.). For even greater formulation of context, the metadata, taxonomy, and ontology specifications for the SHARE framework discussed above will provide domain-specific semantics that should enable more precise discernment of relevance in the searches. As the formalized semantics of the component specification and ontology are developed, the formalisms will be provided to the ReSEARCH developers to determine if improvements in search precision can be achieved.

D. Summary

Enriched semantic specification of the assets in the SHARE repository will enable users to more readily find resources that meet their need in their context. Extensive work in the Web community is providing tools and techniques that can be applied to the SHARE framework. We will select and apply appropriate techniques to meet the goals of the framework development.

VII. Share Framework Development Approach

A. Introduction

Based on our vision for the framework and the related existing technologies we have summarized, in this section we lay out our intended path forward for developing the SHARE repository framework.

B. SHARE Metadata

An initial list of required asset information has been developed by the SHARE Program Office at Naval Surface Warfare Center, Dahlgren, VA. For our research, we will begin by developing a schema based on this initial list and complement the metadata fields with necessary information for filling out the framework. To fill out the data set, we will evaluate known good metadata examples, and we will pull relevant information into the SHARE metadata. We will then ensure that the metadata includes all necessary information to place the artifact in the appropriate context based on the ontology. In order to promote maximum exposure of SHARE contents, we will also ensure that minimum requirements of DDMS are satisfied. Based on all of these considerations, we will develop a practical metadata schema. This will most likely include a core data set and variations for different types of artifacts.

In order to evaluate the completeness of the metadata, we intend to investigate case studies for each phase of the software development cycle. As stated previously, repository user needs vary greatly depending on what the user's needs are at the time of search. Therefore, we are constructing case studies that capture the potential needs based on the user's current development activities. For each of these case studies, the metadata will be evaluated to ensure inclusion of all appropriate information for enabling retrieval decisions.

C. SHARE Software Behavior Representation

For the SHARE software behavior representation, we suspect the overall goal of implementing formalized representations of software behavior, which are standardized across all systems, is not feasible in the short term. While we intend to keep the loftier goal in mind, it is likely that an interim step towards standardization of formal software behavior representation will be required.

One near-term solution may be to use available domain information that standardizes descriptions of software functionality. For example, the Common Systems Function List (CSFL), Common Operational Activities List (COAL), and Common Information Element List (CIEL) are leadership-endorsed listings of combat system functionality that can be utilized as an initial characterization of software behavior. We will investigate the use of a subset of these listings in the development of taxonomies for the SHARE repository framework. If we require asset submitters to state the functionality of the components in these terms, we can then build the tools to guide users in selecting desired behavior in the same terms. We will also explore characterization of software assets based on current and emerging Web Services (e.g., WSDL) and Semantic Web Services (e.g., WS-BPEL, OWL-S) approaches.

D. SHARE Relationship Framework (Ontology)

The ontology for SHARE will be based on several types of relationships between the items in the repository and each other, as well as with relevant domain architectural descriptions and other information. The types of relationships we are exploring are the artifact's place in the software engineering lifecycle, its architectural fit in its original system, its architectural fit in any systems in which it was subsequently used, identification of the component's fit in the Surface Navy Open Architecture reference architecture, and the semantic relationships of various documents in the repository (based on the ReSEARCH work). Each type of relationship will be examined to determine its appropriate representation form (RDF, OWL, Rules, etc.). The goal is to determine representation forms that will best

enable tool development supporting the types of searches described in the previous chapters based on the ontology provided.

VIII. Future Work

Current research will describe the component specification and ontology desired for the SHARE repository. Further work will be necessary to implement the framework and develop a tool suite that will enable the described search capabilities. In the SHARE implementation, additional repository features can be added, such as an Amazon-like “similar results” feature that points people with similar problems to the retrieval of the same files (and other similar recommendations found in Johnson (2007). In the long term, further work will be required if the intent is to eventually enable automated composition of a system based on reusable components. As mentioned previously, a starting point to accomplish this goal may be to standardize a formal behavior representation of the repository contents.

THIS PAGE INTENTIONALLY LEFT BLANK

IX. Summary

This research will result in a component specification and ontology designed to support a tool suite for enabling advanced search and discovery solutions supporting reuse of repository artifacts for every phase of the software lifecycle. We have provided an overview of our intended framework, discussed relevant related technologies and initiatives, and laid out our plan for completing the repository development. We also discussed some possibilities for future work beyond the scope of this initial project.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

Alesso, H. P., & Smith, C. F. (2006). *Thinking on the web: Berners-Lee, Gödel, and Turing*. Hoboken, NJ: John Wiley & Sons, Inc.

Ambler, S. (2007). *UML2 Communication Diagramming Guidelines.* Retrieved March 15, 2007, from http://www.agilemodeling.com/style/collaborationDiagram.htm

Ambler, S. (2006). *Introduction to UML2 State Machine Diagrams..* Retrieved March 15, 2007, from http://www.agilemodeling.com/artifacts/stateMachineDiagram.htm

Bass, L., Clements, P., & Kazman, R. (2003). *Software architecture in practice (2nd ed.).* Boston: Addison-Wesley.

Bell, D. (2004). IBM: UML's sequence diagram. Retrieved March 15, 2007, from, http://www.ibm.com/developerworks/rational/library/3101.html

Clements, P. (1996). A survey of architecture description languages. Proceedings from the 8th International Workshop on Software Specification and Design, Schloss Velen, Germany, 16-25.

Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Nord, R., & Stafford, J. (2003). *Documenting software architectures: Views and beyond*. Boston: Addison-Wesley.

Comprehensive PERL Archive Network (CPAN). (2007). CPAN. Retrieved January 16, 2007, from http://www.cpan.org

Daconta, M. C., Obrst, L. J., & Smith, K. T. (2003). *The semantic web: A guide to the future of xml, web services, and knowledge management*. Indianapolis: Wiley Publishing, Inc.

Department of Defense Chief Information Officer (2003, May 9). Net-centric data sharing strategy. Washington DC

Deputy Assistant Secretary of Defense (2007). Department of Defense Discovery Metadata Specification (DDMS) (Version 1.4.1). Deputy Chief Information Office.

Eastlake III, D. E., & Niles, K. (2003). *Secure XML: The New Syntax for Signatures and Encryption*. Boston: Addison-Wesley.

Erl, T. (2005). Service-oriented architecture: Concepts, technology, and design. Upper Saddle River: Pearson Education, Inc.

Frankel, D. S. (2003). *Model driven architecture: Applying MDA to enterprise computing*. Indianapolis: Wiley Publishing, Inc.

Gordon-Schlosberg, A. (2007) *Palantir screenshots in the wild: Swing Sightings*. Retrieved February 12, 2008, from http://blog.palantirtech.com/2007/09/11/palantir-screenshots/.

Hludzinski, B. (1998, August). Understanding interface definition language:A developer's survival guide. Retrieved March 7, 2008, from http://www.microsoft.com/msj/0898/idl/idl.aspx

Jackson, D. (2006). *Software abstractions.* Boston: MIT Press.

Jia, X., Ying, S., Zhang, T., Cao, H., & Xie, D. (2007). A new architecture description language for service-oriented architecture. Proceedings from the 6th International Conference on Grid and Cooperative Computing, Washington DC, 96-103.

Johnson, J. (2007, October). SHARE repository component specification: Needs assessment. Technical Report, Monterey, CA: Naval Postgraduate School

Larman, C. (2005). *Applying UML and patterns: An introduction to object-oriented analysis and design and iterative development (3rd ed.).* Upper Saddle River: Prentice Hall.

Medvidovic, N., & Taylor, R. (2000). A classification and comparison framework for software architecture description languages. *IEEE Transactions on Software Engineering*, *26*(1),70-93.

Michael, B. (2008, March 19). Perspectives, practices, and the future of building highly dependable and trustworthy systems. Software Engineering Presentation. Naval Postgraduate School, Monterey, CA.

Object Management Group. (2005). *Reusable asset specification, Version 2.2.* Retrieved January 29, 2008, from http://www.omg.org/technology/documents/formal/ras.htm

Object Management Group. (2007). OMG IDL: Details. Retrieved February 22, 2008, from http://www.omg.org/gettingstarted/omg_idl.htm

Simventions (2008, January 22). Modeling and simulation (M&S) community of interest (COI) discovery metadata specification (MSC-DMS) (Version 1.0.1). Washington, DC: DoD Modeling and Simulation Coordination Office (M&S CO).

Sarkar, M., & Brown, H. (1993). Graphical fisheye views. *Communications of the ACM*, *37*, 73-83.

Shaw, M., & Clements, P. (2006). The golden age of software architecture. *IEEE Software, 23*(2), 31-39.

Spivey, J. (1992). *The Z NOTATION: A reference manual (2nd ed.).* Oxford:Prentice Hall.

SourceForge. (2007). *SourceForge.net: Welcome to SourceForge.net.* Retrieved October 8, 2007, from www.sourceforge.net

Szyperski, C. (2002). *Component software: Beyond object-oriented programming, (2nd ed.)*. New York: Addison-Wesley.

World Wide Web Consortium (W3C). (1994). *Semantic Web Stack.* Retrieved February 26, 2008, from www.w3.org/2007/03/layerCake.png

WordNet. (2006). A lexical database for the English language. Retrieved March 25, 2007, from http://wordnet.princeton.edu/

Zhang, B., Ding, K., & Li, J. (2001). An XML-message based architecture description language and architectural mismatch checking. Proceedings from the 25th Annual International Computer Software and Applications Conference, Washington, DC, 561-566.

THIS PAGE INTENTIONALLY LEFT BLANK

2003 - 2008 Sponsored Research Topics

Acquisition Management

- Software Requirements for OA
- Managing Services Supply Chain
- Acquiring Combat Capability via Public-Private Partnerships (PPPs)
- Knowledge Value Added (KVA) + Real Options (RO) Applied to Shipyard Planning Processes
- Portfolio Optimization via KVA + RO
- MOSA Contracting Implications
- Strategy for Defense Acquisition Research
- Spiral Development
- BCA: Contractor vs. Organic Growth

Contract Management

- USAF IT Commodity Council
- Contractors in 21st Century Combat Zone
- Joint Contingency Contracting
- Navy Contract Writing Guide
- Commodity Sourcing Strategies
- Past Performance in Source Selection
- USMC Contingency Contracting
- Transforming DoD Contract Closeout
- Model for Optimizing Contingency Contracting Planning and Execution

Financial Management

- PPPs and Government Financing
- Energy Saving Contracts/DoD Mobile Assets
- Capital Budgeting for DoD
- Financing DoD Budget via PPPs
- ROI of Information Warfare Systems
- Acquisitions via leasing: MPS case
- Special Termination Liability in MDAPs

Human Resources

- Learning Management Systems
- Tuition Assistance
- Retention
- Indefinite Reenlistment
- Individual Augmentation

Logistics Management

- R-TOC Aegis Microwave Power Tubes
- Privatization-NOSL/NAWCI
- Army LOG MOD
- PBL (4)
- Contractors Supporting Military Operations
- RFID (4)
- Strategic Sourcing
- ASDS Product Support Analysis
- Analysis of LAV Depot Maintenance
- Diffusion/Variability on Vendor Performance Evaluation
- Optimizing CIWS Lifecycle Support (LCS)

Program Management

- Building Collaborative Capacity
- Knowledge, Responsibilities and Decision Rights in MDAPs
- KVA Applied to Aegis and SSDS
- Business Process Reengineering (BPR) for LCS Mission Module Acquisition
- Terminating Your Own Program
- Collaborative IT Tools Leveraging Competence

A complete listing and electronic copies of published research are available on our website: www.acquisitionresearch.org

Initial Distribution List

1. Defense Technical Information Center — 2
   8725 John J. Kingman Rd., STE 0944; Ft. Belvoir, VA 22060-6218
2. Dudley Knox Library, Code 013 — 2
   Naval Postgraduate School, Monterey, CA 93943-5100
3. Research Office, Code 09 — 1
   Naval Postgraduate School, Monterey, CA 93943-5138
4. William R. Gates — 1
   Dean, GSBPP
   Naval Postgraduate School, Monterey, CA 93943-5138
5. Stephen Mehay — 1
   Associate Dean for Research, GB
   Naval Postgraduate School, Monterey, CA 93943-5138
6. Jean Johnson — 1
   Research Assistant, Systems Engineering Department
   Naval Postgraduate School, Monterey, CA 93943-5138
7. Curt Blais — 1
   Research Associate, MOVES Institute
   Naval Postgraduate School, Monterey, CA 93943-5138

Copies of the Acquisition Sponsored Research Reports may be printed from our web site www.acquisitionresearch.org